Claims

Please amend the claims as follows:

1. (previously presented) A computer readable medium having stored thereon a computer executable compiler system that performs semantic analysis of interface definition language constructs embedded in programming language code in a file, the compiler system comprising:

a front end module that separates a file into plural tokens, the file including programming language code having embedded therein interface definition language constructs;

a converter module that converts the plural tokens into an intermediate representation, wherein the intermediate representation includes a symbol table and a tree that unifies representation of the programming language code and the embedded interface definition language constructs, wherein at least some of the embedded interface definition language constructs are represented in the tree without creating new programming language code for the at least some of the embedded interface definition language constructs, wherein the symbol table includes plural entries for symbol names for the programming language code, and wherein at least one of the plural entries has an associated list of definition language attributes; and

a back end module that produces output computer-executable code from the intermediate representation based at least in part upon semantics of the embedded interface definition language constructs.

2-3. (cancelled)

4. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the compiler system further ~~comprising~~ comprises a definition language attribute provider that modifies the intermediate representation based upon the semantics of the embedded interface definition language constructs.

5. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the compiler system further ~~comprising~~ comprises an error checker module that checks for lexical, syntactic, and semantic errors in the file.

6. (previously presented) A computer executable compiler system stored in a computer system that creates a unified programming language and interface definition language parse tree from a file comprising a mix of programming language constructs and interface definition language constructs, the compiler system comprising:

a front end module that separates a file into plural tokens, the file comprising a mix of programming language constructs and interface definition language constructs; and

a converter module that converts the plural tokens into an intermediate representation comprising a symbol table and a parse tree, wherein:

the symbol table includes plural entries for symbol names for the programming language constructs, at least one of the plural entries having an associated list of interface definition language attributes;

the parse tree unifies representation of the programming language constructs and the interface definition language constructs; and

at least some of the interface definition language constructs are represented in the parse tree without creating new programming language constructs for the at least some of the interface definition language constructs.

7. (original) The compiler system of claim 6 wherein the front end module recognizes a delimiting character that distinguishes interface definition language tokens from programming language tokens.

8. (original) The compiler system of claim 6 further comprising an error checker module that performs lexical and syntactic checks on the file.

9. (previously presented) A computer readable medium having stored thereon a data structure representing a unified interface definition language and programming language parse tree for a file having a combination of programming language code and embedded interface definition language constructs, the data structure comprising:

a first data field storing data representing a symbol table that has plural entries, each of the plural entries corresponding to a symbol name for programming language code of a file having a combination of programming language code and embedded interface definition

language constructs, at least one of the plural entries having an associated list of interface definition language attributes based upon the embedded interface definition language constructs; and

a second data field storing data representing a parse tree, wherein the parse tree unifies representation of the programming language code and the embedded interface definition language constructs; and

wherein at least some of the embedded interface definition language constructs are represented in the parse tree without creating new programming language code for the at least some of the embedded interface definition language constructs.

10-15. (canceled)

16. (previously presented) A computer readable medium having stored thereon instructions for performing a method of creating a unified programming language and definition language tree from a file that includes interface definition language constructs embedded in programming language code, the method comprising:

separating a file into plural tokens, the file including interface definition language constructs embedded in programming language code;

building a symbol table having plural entries for symbol names for the programming language code, at least one of the plural entries having an associated list of definition language attributes based upon the embedded interface definition language constructs; and

building a tree that unifies representation of the embedded interface definition language constructs and the programming language code, wherein the building comprises representing at least some of the embedded interface definition language constructs in the tree without creating new programming language code for the at least some of the embedded interface definition language constructs.

17. (previously presented) The computer readable medium of claim 16 wherein the separating comprises recognizing a delimiting character that distinguishes definition language tokens from programming language tokens.

18-21. (canceled)

22. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the backend module also produces output interface definition language information in an output file that includes the output computer-executable code.

23. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the backend module also produces output interface definition language information in a separate output file from the output computer-executable code.

24. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the output computer-executable code is computer-executable instructions for a real processor.

25. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the output computer-executable code is computer-executable instructions for a virtual processor.

26. (currently amended) The ~~compiler system~~ computer readable medium of claim 1 wherein the programming language code is in C++ and wherein the embedded interface definition language constructs includes IDL constructs.

27-36. (canceled)

37. (currently amended) The ~~method~~ computer readable medium of claim 1 wherein the embedded interface definition language constructs ~~includes~~ include an export attribute, wherein the export attribute annotates a user-defined data type, and wherein the compiler system outputs interface definition language metadata for the user-defined data type based at least in part upon the export attribute.

38. (currently amended) The ~~method~~ computer readable medium of claim 1 wherein the embedded interface definition language constructs ~~includes~~ include an interface type attribute, wherein the interface type attribute annotates an interface, and wherein the output computer-executable code includes implementation code for an implementation of the interface.

39. (currently amended) The ~~method~~ computer readable medium of claim 38 wherein the interface is a standard COM interface, dispatch interface, or dual interface, and wherein the implementation code is for an object that exposes the interface.

40. (currently amended) The ~~method~~ computer readable medium of claim 38 wherein the back end module uses directional attributes for arguments of member functions of the interface to produce the implementation code, and wherein the directional attributes include one or more of in, out, and retval.

41. (currently amended) The ~~method~~ computer readable medium of claim 1 wherein the embedded interface definition language constructs ~~includes~~ include a project attribute.

42. (currently amended) The ~~method~~ computer readable medium of claim 1 wherein a definition language attribute provider reacts to plural events during compilation of the file by causing modification of the intermediate representation, wherein at least one of the plural events, in reaction to which the definition language attribute provider causes modification of the intermediate representation, occurs during processing of the embedded interface definition language constructs, and wherein at least one of the plural events, in reaction to which the definition language attribute provider causes modification of the intermediate representation, occurs during processing of the programming language code.